SDEP prototype

Why do we need to test APIs?

1. APIs define the business logic and rules for user interaction with services and data.
2. API monitoring in production ensures live endpoints are functioning properly and delivering expected data.
3. API testing ensures complete coverage and validates various user application manipulations.
4. The recommended approach is to conduct both UI and API testing for comprehensive testing coverage.
5. Early detection and resolution of issues prevent customer dissatisfaction and maintain accessibility.
6. UI testing alone is insufficient as it focuses on the presentation layer and may miss critical scenarios.

Benefits of using API testing in the development phase

Test Quality

API testing allows for focus on performance in failure scenarios and helps identify how the system handles errors, improving the end user experience. Iterating on API tests enhances test coverage and ensures comprehensive testing.

Test Coverage

Conducting API testing during development uncovers issues with the API server, other services, and network that may not be easily identified post-deployment.

Test Reuse

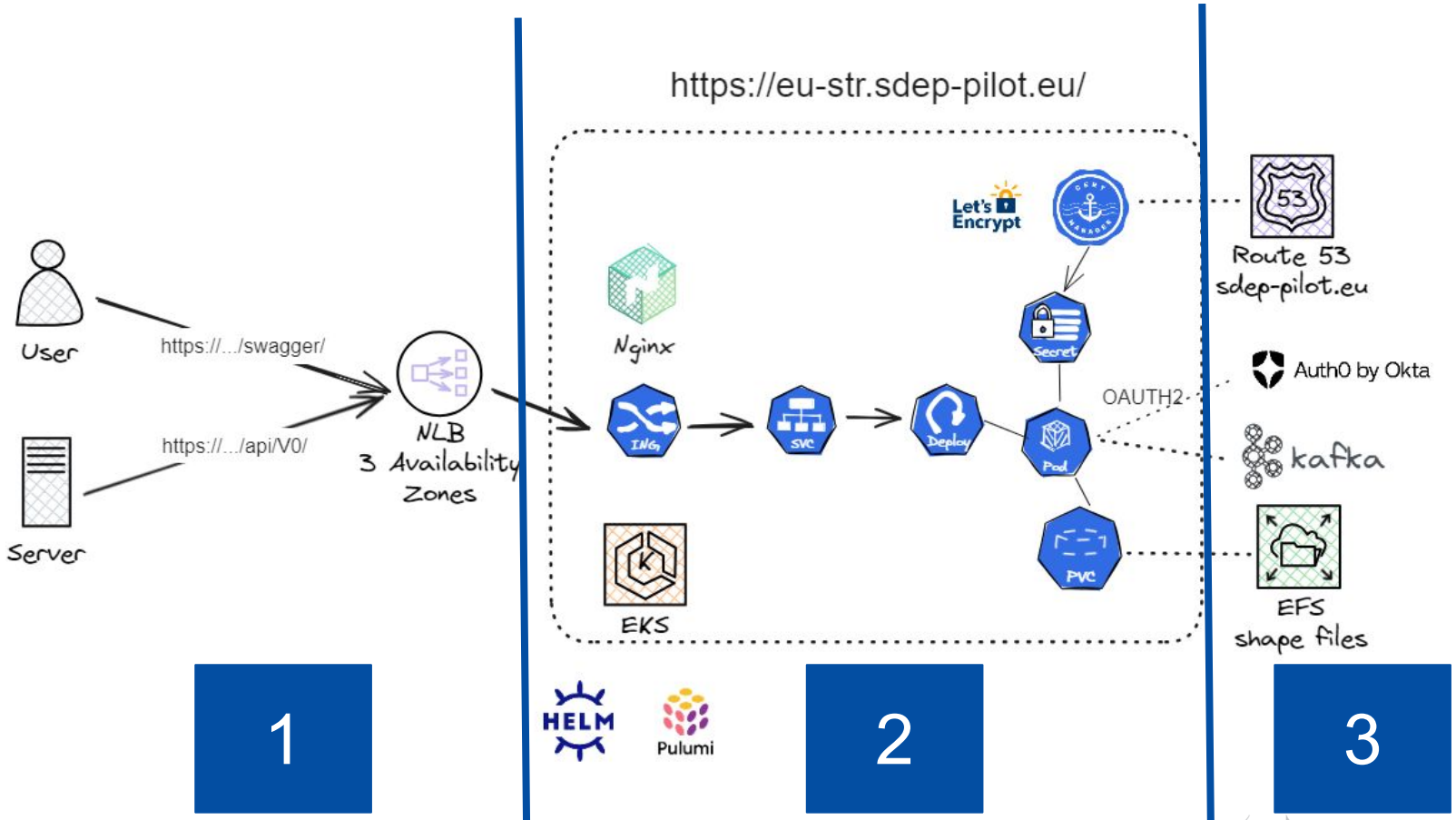
API tests built during development provide better coverage and prevent service issues in production.

Architecture

In the core, It is important to maintain vendor neutrality.

Components around can be changed.

Integration is achieved using general standards.



Architecture

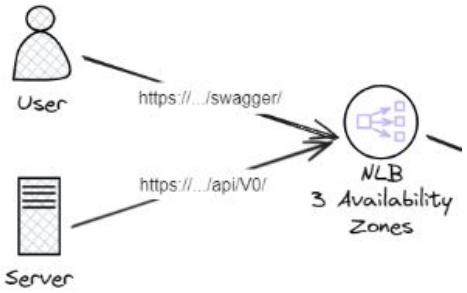
User/Server Requests:

- **User**: Accesses the API via a web interface using `https://.../swagger/`.
- **Server**: Makes API calls directly using `https://.../api/v0/`.

1

Network Load Balancer (NLB):

- The requests from users and servers are directed to the Network Load Balancer (NLB), which distributes the incoming traffic across multiple availability zones. The NLB is a critical component in our architecture. Its primary role is to distribute incoming traffic across multiple backend servers to ensure that no single server becomes overwhelmed with requests. This helps maintain high availability and reliability of the service.



European Commission

Architecture

2

Nginx Ingress Controller:

- The NLB forwards the traffic to the Nginx Ingress Controller, which manages external access to the services in the Kubernetes (EKS) cluster, handling routing, load balancing and can also do rate limits

Kubernetes Service:

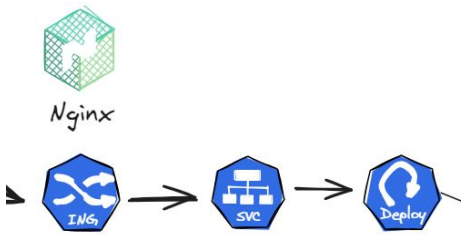
- The Ingress Controller forwards the requests to the appropriate Kubernetes Service, which acts as a bridge between the external requests and the internal Pods.

Deployment :

- The service forwards the requests to the appropriate Deployment, which ensures that the desired number of pod replicas are running and available to handle the requests.

Elastic Kubernetes Service:

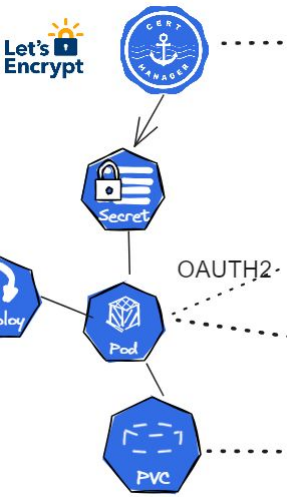
- The entire deployment is managed within an Amazon EKS cluster, which provides a managed Kubernetes environment to run the containers.



European Commission

Architecture

2



Pod:

- The Deployment manages Pods, which are the smallest deployable units in Kubernetes. Pods encapsulate the application containers and their resources.

Secrets:

- Kubernetes Secrets are used to manage sensitive information, such as API keys, passwords, and certificates, required by the Pods.

Let's Encrypt:

- Let's Encrypt is used to obtain SSL/TLS certificates to secure the communication between the clients and the services.

Persistent Volume Claim (PVC):

- PVCs are used to request storage resources in the EKS cluster. They allow the Pods to persist data, even after the Pods are destroyed and recreated.

Architecture

2

- **Helm**: A package manager for Kubernetes, used to define, install, and upgrade even the most complex Kubernetes applications.

- **Pulumi**: An infrastructure as code tool that allows defining infrastructure using programming languages, used to provision and manage the infrastructure resources in this setup.

HELM

Pulumi

European Commission

Architecture

Route 53:

- AWS Route 53 provides DNS services, ensuring that the requests to the API are routed correctly to the NLB.

Auth0 by Okta:

- Auth0, managed by Okta, handles OAuth2 authentication, ensuring secure access to the API services by verifying the identity of the users.

Kafka:

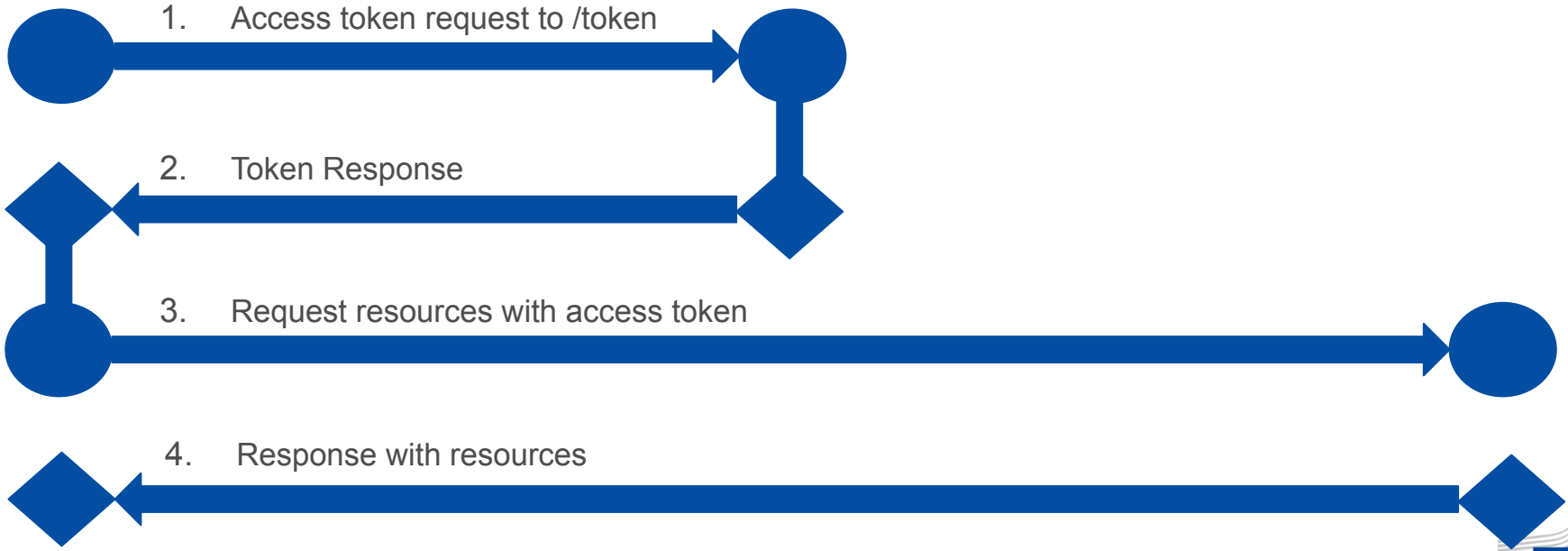
- Apache Kafka is used for event streaming, allowing the services to publish and subscribe to streams of records (Activity Data), enabling real-time data processing and integration between services.

EFS (Elastic File System):

- AWS EFS provides scalable file storage that can be mounted to the Pods. It is used to store and share files, such as shape files, across multiple Pods.

Authorization flow

Client-Side	Auth Server	Server-side
Your Application	Auth0	Resource server



Tools commonly used for API Testing

The most commonly used tools to test RESTful web services are as follows:

• **Curl** - Curl is a command-line tool for transferring data with URLs. It is commonly used for testing RESTful web services by making HTTP requests and reading their responses. Curl is very powerful and flexible, allowing users to perform a wide range of tasks such as sending HTTP GET, POST, PUT, DELETE requests, and handling various types of authentication and data formats.

• **Postman** - Being originally a Chrome browser plugin, Postman now extends their solution with the native version for both Mac and Windows. It is a popular API client that makes it easy for developers to create, share, test and document APIs. This is done by allowing users/testers to create and save simple and complex HTTP/s requests, as well as read their responses

• **SwaggerUI** - Swagger is the largest framework for designing APIs using a common language and enabling the development across the whole API lifecycle, including documentation, design, testing, and deployment. It allows testers to validate your APIs on the cloud , i.e. without any kind of set-ups or desktop downloads.

Thank you

European Commission